In the *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems*, Charlotte, NC, 1991, pages 358-367.

IMPROVING THE COMPREHENSIBILITY, ACCURACY, AND GENERALITY OF REACTIVE PLANS

Diana F. Gordon
Navy Center for Applied Research in AI
Naval Research Laboratory, Code 5510
Washington, D.C. 20375-5000

Abstract

This paper describes a method for improving the comprehensibility, accuracy, and generality of reactive plans. A reactive plan is a set of reactive rules. Our method involves two phases: (1) formulate explanations of execution traces, and (2) generate new reactive rules from the explanations. The explanation phase involves translating the execution trace of a reactive planner into an abstract language, and then using Explanation Based Learning to identify general *strategies* within the abstract trace. The rule generation phase consists of taking a subset of the explanations and using these explanations to generate a set of new reactive rules to add to the original set for the purpose of performance improvement.

1. Introduction

Reactive planning has proven to be a highly effective approach to planning (e.g., [10]). We have developed an enhancer for reactive plans that satisfies two goals. The first goal is to facilitate human understanding of plans generated by a particular class of reactive planners. This class consists of planners that gain effectiveness at the expense of human comprehensibility (e.g., [1] and [4]). The second goal is to improve the reliability and generality of reactive plans.

Figure 1 diagrams our approach which is divided into two phases, each associated with one of the goals. An *explanation phase* appears to the left of the dotted line in Figure 1 and a *rule generation phase* appears to the right. The explanation phase begins with the application of a reactive planning system to a problem. The system generates an execution trace, which is translated into an abstract language trace. Next, a variant of Explanation Based Learning

(EBL) is applied to explain the abstract trace. EBL is a knowledge-intensive technique for forming a general explanation from a specific example [9]. EBL assumes that there exists a previously constructed domain theory and uses this theory to deductively derive explanations. The explanation phase of our method, which captures generalizations such as symmetries, culminates in English text describing the high level strategies of the reactive planner. We assume that each plan is composed of a set of reactive rules. During the rule generation phase, a subset of the explanations from the previous phase is used to generate a set of new reactive rules to add to the original plan. The new rules behave as a *local expert* because they remedy localized weaknesses of the original plan. The system may now use this enhanced plan to tackle the problem again. Our implementation of this method is called EXplain-and-GENerate (EXGEN).

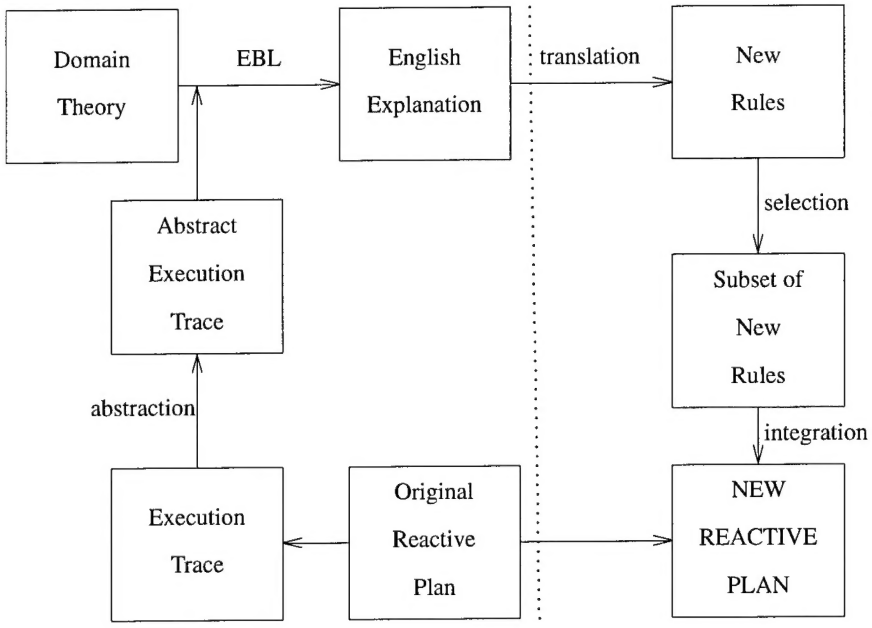

FIG. 1. Plan explanation and rule generation.

We have applied this approach to a reactive plan and a problem described in Section 2. For this problem, the reactive planner's knowledge is imperfect. We know this because the planner, when using this knowledge (the original reactive plan), has less than a perfect success rate. Furthermore, the domain theory used to explain and generate rules is only complete for a local area of expertise. What is most interesting is that both sources of knowledge complement each other, one filling in the knowledge gaps for the other. When EXGEN is added to the reactive planner, the newly generated set of rules helps the planner to achieve 100% success.

Previous research is related to this work. EXGEN is similar to Learning Apprentice Systems (e.g., [3], [12]) because it uses EBL to explain the behavior of an expert and then uses

this explanation to generate new planning rules. The difference is that Learning Apprentice Systems learn from a human to improve a system, whereas EXGEN learns from a system for the purposes of human understanding and system improvement. The type of EBL invoked by EXGEN is *Plausible EBL*, which has been developed by DeJong and Gervasio [3] for deriving plausible (i.e., uncertain and imprecise but generally true) explanations of events that occur in continuous domains. These explanations are expressed in the language of Qualitative Process Theory [2]. In terms of completing the gaps in a domain theory, several researchers provide outstanding examples ([6], [12], and [13]). However, we do not employ any of these approaches because their assumptions are not met in our situation. Instead, we use the original reactive plan to fill the knowledge gaps in the domain theory. The translation from high-level explanations to reactive rules is related to two techniques: specialization [8] and symbol grounding [7].

Despite these similarities to previous research, our approach is novel. Reactive planners are becoming increasingly complex in terms of their behavior. In response to this growing complexity, we have developed a plan enhancer. Ours is the first system that can explain the behavior of a reactive planner in human-oriented form, then improve the reactive plan based on the explanation. Since most people do not fully trust automation, especially if lives are at stake, people will probably want to screen system-generated plans for many applications. A plan enhancer can increase the acceptability of reactive plans to humans.

2. The Reactive Planner and Domain

So far, we have tested our method with the SAMUEL system [5]. This system uses a genetic algorithm and other competition-based heuristics to learn high performance reactive plans in the absence of a strong domain theory. SAMUEL has three major components: a problem specific module, a performance module, and a learning module. In this research, we use a plan that has already been learned by the system. Therefore, our method only employs the problem specific and performance modules. Although reactive planning involves both learning and executing a plan, for simplicity, we use the terms ''SAMUEL'' and ''reactive planner'' to refer to the performance module of this system.

SAMUEL has been applied to a variety of domains, such as Evasive Maneuvers (EM), Tracking, and Dogfighting. We would like to see if EXGEN can improve this system's performance in each of these domains. Schultz and Grefenstette [11] have demonstrated that the addition of manually developed rules can improve SAMUEL's performance for the Evasive Maneuvers problem. Since manually developed rules have improved the performance on EM, we have started by testing our approach to automatic rule generation on this problem.

In EM, which is simulated two-dimensionally, an agent (controlled by the reactive plan) tries to evade an adversary. The adversary tracks and tries to destroy the agent. We divide EM into *episodes* that begin with the adversary approaching the agent from a randomly chosen direction and that end when either the agent is destroyed by the adversary or the adversary's speed falls below a given threshold. The latter occurs because, although the adversary's speed is initially greater than the agent's speed, the adversary loses speed as it chases the agent. Six sensors provide information about the current state: the agent's *last-turn*, the *time*, and the adversary's *range*, *bearing*, *heading*, and *speed*. One action variable controls the agent, namely, the agent's *turning-rate* (also called *turn*).

An example of an actual reactive rule learned by SAMUEL for EM is the following:

```
IF    (and (last-turn [-135, 135]) (time [2, 12]) (range [0, 700])
           (bearing [2, 6]) (heading [0, 30]) (speed [100, 950]))
THEN  (turn 90)
```

where ''[*X, Y*]'' denotes sensor values from *X* through *Y*. For example, ''(last-turn [-135, 135])'' matches if the last turn that the agent made was between -135 and 135 degrees. The action recommended by this rule is to turn 90 degrees. Although individual rules are understandable, the general strategy underlying a chain of rule firings is not. The strategies are cryptic because SAMUEL uses complex methods, such as conflict resolution and partial matching, when executing a reactive plan. Furthermore, these rules do not contain information about subgoals, such as ''increasing range of adversary'', that the rules are designed to achieve.

3. Explanation Phase

Our method begins by generating explanations of execution traces. An execution trace is composed of a chronological sequence of snapshots of numeric sensor readings resulting from a sequence of actions. In EM, one trace corresponds to one episode, and the success or failure of the agent at evading the adversary is noted at the end of the trace. Currently, only *success traces* are explained (i.e., traces where the agent evades the adversary). Using a predefined quantitative-to-qualitative mapping, each execution trace is translated into an abstract trace. The abstract trace contains high-level qualitative terms, such as ''far'' or ''behind'', in place of numeric values, as well as first and second derivative information.

After translating to abstract terms, EXGEN uses (Plausible) EBL, which requires a previously developed domain theory. To develop a domain theory for the EM problem, we have

run a program that calculates the average frequencies of qualitative values in 100 traces of previous episodes. This domain theory is used by EBL to identify plausible *strategies* within the abstract trace. The most frequently occurring values are considered to be the most plausible elements of a strategy. A strategy consists of a set of *triggering preconditions* (i.e., sensor readings) and a *triggering action* (which may be an action derivative) followed by a chain of causal events that ultimately results in the satisfaction of a subgoal. The triggering preconditions must be present in order to fire a triggering action. A triggering action is directly controllable by the agent, whereas a subgoal is not. A strategy is equivalent to an EBL proof. An example of an EBL proof for EM is given below. In this proof, (Q+ *X Y*) means that parameter *X* is directly proportional to parameter *Y*.

(P1) (EXPLAIN (INCREASING-ADVERSARY-DECELERATION)
(Q+ ADVERSARY-DECELERATION ADVERSARY-TURNING-RATE)
(Q+ ADVERSARY-TURNING-RATE AGENT-TURNING-RATE)
(INCREASING AGENT-TURNING-RATE))
Preconditions: {ADVERSARY-RANGE=NOT-FAR, ADVERSARY-SPEED=HIGH}

The subgoal of P1 is ''increasing adversary deceleration'', and the triggering action is ''increasing agent turning rate''.

The presence of triggering preconditions and actions from the proof are verified in the execution trace. If present, a verified strategy has been found. If not present, the next shortest EBL proof is generated (if possible). If no more proofs can be generated for a subgoal, an alternative subgoal is tried. Whenever a verified strategy is found, the system performs a largely cosmetic translation of that strategy to an English explanation. An explanation that has been generated from P1 is:

(E1) The triggering action of the agent, which is increasing turning rate, caused the adversary's turning rate to have value increasing, which caused the adversary's deceleration to have value increasing, which caused the subgoal, increasing adversary deceleration, to have been achieved between times 2 and 3. The triggering preconditions are: (1) the adversary's range is not far, and (2) the adversary's speed is high.

Explanation E1 and others that are generated by EXGEN clarify the underlying motivations of a complex reactive planner. A more detailed description of the explanation phase may be found in [4].

4. Rule Generation Phase

During the rule generation phase, the system generates new reactive rules from explanations. Each rule is formed from the triggering preconditions and action of the strategy associated with an explanation. The rule is expected to achieve the subgoal for which it is a trigger. The mapping that is used to convert from abstract to concrete numeric values is the inverse of the mapping from concrete to abstract. Multiple rules are generated from a single explanation. One rule that is generated from explanation E1 is:

(R1) IF (and (last-turn [45, 45]) (range [0, 500]) (speed [400,1000]))
THEN (turn 135)

This rule will increase the turning rate of the agent because the ''last-turn'' is 45 degrees and the next ''turn'' is 135 degrees.

An important part (which is partially implemented) of the rule generation phase is the selection of those new rules that can improve SAMUEL's performance. The reader should be aware during the remaining discussions that the way SAMUEL uses a reactive plan is highly complex due to partial rule matching and other features. Therefore, we do not have a clear idea of precisely how the new rules patch the gaps in the original rule set. Instead, we identify gaps in the original plan by analyzing planning failures that appear in the execution trace.

We have decided to add the newly generated rules to the original rule set, rather than to use them alone, because our domain theory is incomplete. Instead of consulting a domain expert to complete the theory, we adopt a complementary empirical-analytical approach to the problem. The original rule set, called ''rules-sam'', yields partial expertise and the new rules yield partial expertise.

How can this complementary blend of old and new rules be implemented? Experiments have indicated that an integration problem arises if this blend is performed arbitrarily. In other words, the new rules can force SAMUEL to enter parts of the search space for which no new rule applies and the rules from rules-sam do not provide adequate expertise. To illustrate with a specific EM scenario, suppose the actions taken at time T and $(T + 1)$ using rules-sam are ''turn right'' and then ''turn left'' respectively. If the explanation process captures symmetry, then the action recommended by a new rule at time T might be ''turn left''. If the domain theory is incomplete, then perhaps no new rule would apply at $(T + 1)$. If rules-sam also lacks knowledge for the new situation, then SAMUEL might apply the original rule that fired at $(T + 1)$, which has the action ''turn left''. The pair of ''turn left'' actions at times T and $(T + 1)$ would not perform the original direction reversal that was probably effective.

Therefore, instead of arbitrarily blending rules, we have chosen to decompose the main problem (e.g., EM) into two stages. A local expert is used to solve the latter stage. This is similar to the development of an expert to solve chess end games. The reason for developing an expert for the latter stage is that once control is transferred to this local expert, it will not return to rules-sam. Rule integration is therefore not a problem. The local expert that we have developed, called ''rules-expert'', consists of a subset of the new rules.

For EM, we have identified a latter stage of this problem, and EXGEN has derived the corresponding local expert. There are 3 natural ways to bisect an EM episode: 1) find a threshold value for the adversary's speed, e.g. the subproblem occurs when speed ≤ 250, or 2) find a threshold value for time, or 3) find a relationship between the adversary and agent's speeds. We have decided to use the third option because it is the most general, i.e. it does not involve specific constants. To discover this relationship, we have added a sensor *agent's speed* to SAMUEL's planner. Execution traces with the new sensor reveal that rules-sam fails only when the adversary's speed is equal to the agent's speed. To avoid integration problems, though, we need to define a stage that extends to the end of an EM episode. Fortunately, we know that once the adversary's speed goes below the agent's speed, the adversary's speed remains below it. The subproblem called *restricted EM*, in which the adversary's speed is less than or equal to the agent's speed, satisfies the constraint that control need not be returned to rules-sam. Therefore, we need a local expert designed for restricted EM.

Out of all the explanations generated by EXGEN, experiments have determined that two explanations suffice to generate this local expert. The first states that if the agent turns away from the adversary then the agent achieves the subgoal ''adversary behind agent'' (called explanation ''E2''). The second states that if the agent moves straight when the adversary is behind then the agent achieves the subgoal ''increasing adversary range'' (called explanation ''E3''). EXGEN converts the triggering preconditions and actions of E2 and E3 into new reactive rules. Rules-expert consists of these new rules.

During the process of creating rules-expert for EM, we have discovered that the qualitative-to-quantitative mapping used in the new rules, which was manually derived, is inadequate for high performance. Therefore, we have run experiments to compare several candidate mappings. Figure 2 highlights the importance of selecting a good mapping. This figure contrasts the performance of rules-expert with 2 different mappings. Each curve represents the average performance over 1000 episodes of SAMUEL using this rule set. Each point on the curves gives performance for a different initial adversary speed. Initial adversary speeds are all less than or equal to the initial agent speed, which is 333. (Recall that during an episode the adversary loses speed but never regains speed.) The greatest performance disparity occurs when the initial adversary speed is 333. In that case, the rules with Mapping

M2 achieve 100% success whereas those with Mapping M1 only achieve 61.2% success.

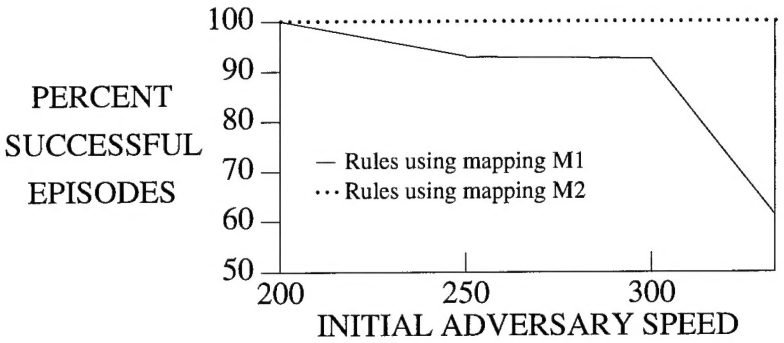

FIG. 2. Comparison of mappings on restricted EM.

The mapping that we have chosen (M2) for the final implementation of rules-expert is empirically derived by a program that averages execution trace values over the same 100 episodes of SAMUEL that were used for generating the EBL domain theory. EXGEN's quantitative-to-qualitative mapping, which was formerly derived manually, is now the inverse of our new mapping. The final rules-expert, with the new mapping, enables the planner to succeed 100% of the time over 1000 episodes of *restricted* EM. Furthermore, the rules in rules-expert are mutually exclusive as well as exhaustive with respect to the ranges of sensor values that they cover.

To adapt the rules in rules-expert to be part of a plan for the original EM problem, EXGEN adds a test for ''adversary speed ≤ agent speed'' to the conditions of the new reactive rules.† Also, we have modified SAMUEL's conflict resolution mechanism to prefer the new EBL rules whenever they apply. Since the rules are mutually exclusive and exhaustive, this implies that once control moves to rules-expert, a deterministic sequence of actions are taken by the agent that never fail. These steps yield a local expert for restricted EM that can fill the knowledge gap of rules-sam. The last step of the rule generation phase is to combine rules-expert with rules-sam to form the final rule set, which is called ''rules-combo''.

The rules-sam that is used for generating execution traces in EM is a high-performing rule set learned by SAMUEL. Using rules-sam, SAMUEL wins (i.e., evades the adversary) 992 out of 1000 episodes (99.2%). However, when rules-expert is added to rules-sam to form rules-combo, the success over 1000 episodes climbs to 100% on the *original* EM problem. Although the performance improvement from 99.2% to 100% gained by adding the new rules

† EXGEN exhaustively generates all necessary combinations of agent and adversary speeds that satisfy this test. Future work will include modifying SAMUEL's rule language to directly express this test.

to rules-sam would be insignificant for many applications, there are some applications for which this would be very meaningful. One example is the life-threatening situation of a human pilot using this plan to evade a missile that is tracking his plane. Furthermore, because they are based on generalized explanations, the new rules are more general than those in rules-sam and are therefore potentially applicable under more varied conditions.

5. Summary and Future Work

The goal of this research has been to improve the acceptability of reactive plans for human use. Two steps are taken toward this end, as described here: (1) human comprehensibility is improved, and (2) reliability and generality are increased. Using the approach that we present, it is possible to increase the comprehensibility and accuracy of reactive plans.

Although comprehensibility is greatly improved by this method, the accuracy improves by only a small margin for the EM problem. This is due to the fact that the original plan developed by SAMUEL is already highly effective. In the future, we would like to apply this method to other problems on which SAMUEL is less effective to see if a greater performance improvement can be achieved. A local expert for more complex domains might be less effective than the one we have developed for restricted EM. For example, it might not complete all of the knowledge gaps of the planner. Nevertheless, even if we cannot achieve a final success rate of 100%, we would like to strive for a large performance improvement on these domains.

When applying EXGEN to other domains, we would also like to evaluate its generality. All of EXGEN's learning methods, other than the separation of EM into stages, seem to be applicable to many other domains. However, the separation of EM into stages might be overly simplistic for other problems. Future work will focus primarily on developing and implementing a more generally applicable method for handling rule integration.

Acknowledgements

I would like to thank John Grefenstette for suggesting the EXGEN project and providing SAMUEL. I would like to thank Bill Spears for his helpful comments, which include suggesting the use of a ''local expert''. I would also like to thank Jude Shavlik for indicating related literature, and members of the Machine Learning Group at NCARAI for proofreading this paper.

References

[1] Brooks, R. (1990). Elephants don't play chess. *Robotics and Autonomous Systems, 6.* Elsevier Publishers.

[2] Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence, 24(1-3).* North-Holland Publishing Company, Amsterdam, The Netherlands.

[3] Gervasio, M. and DeJong, G. (1989). Explanation-based learning of reactive operators. *Proceedings of the Sixth International Workshop on Machine Learning.* Ithaca, NY.

[4] Gordon, D. and Grefenstette, J. (1990). Explanations of empirically derived reactive plans. *Proceedings of the Seventh International Conference on Machine Learning.* Austin, TX.

[5] Grefenstette, J., Ramsey, C. and Schultz, A. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning Journal, 5(4).* Kluwer Academic Publishers, Hingham, MA.

[6] Hall, R. (1986). Learning by failing to explain. *Proceedings of the Fifth National Conference on Artificial Intelligence.* Philadelphia, PA.

[7] Harnad, S. (1990). The symbol grounding problem. *Physica D, 42(1/3).* North Holland Publishers.

[8] Langley, P. (1987). A general theory of discrimination learning. *Production System Models of Learning and Development.* D. Klahr, P. Langley, and R. Neches (eds), The MIT Press, Cambridge, MA.

[9] Mitchell, T., Keller, R. and Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning Journal, 1(1),* Kluwer Academic Publishers, Hingham, MA.

[10] Schoppers, M. (1987). Universal plans for reactive robots in unpredictable environments. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* Milan, Italy.

[11] Schultz, A. and Grefenstette, J. (1990). Improving tactical plans with genetic algorithms. *Proceedings of the Tools for Artificial Intelligence Conference.* Herndon, VA.

[12] Tecuci, G. and Kodratoff, Y. (1990). Apprenticeship learning in imperfect domain theories. *Machine Learning: An Artificial Intelligence Approach, Volume III.* Y. Kodratoff and R. Michalski (eds), Morgan Kaufmann Publishers, Inc.

[13] Wilkins, D. (1990). Knowledge base refinement as improving an incorrect and incomplete domain theory. *Machine Learning: An Artificial Intelligence Approach, Volume III.* Y. Kodratoff and R. Michalski (eds), Morgan Kaufmann Publishers, Inc.